INFO2050 - Programmation avancée

Projet 2: Arbre binaire de recherche

Assistants: Jean-Michel Begon, Jean-François Grailet
Professeur: Pierre Geurts

30 octobre 2015 — édition le 16 novembre (date de remise)

En plus de la recherche d'une clé particulière, il peut être intéressant de pouvoir renvoyer toutes les clés d'un dictionnaire situées dans un certain intervalle de valeurs (défini par deux bornes). Ce type de recherche a de nombreuses applications pratiques. Par exemple, dans le cadre d'un site d'e-commerce, il peut être intéressant de laisser la possibilité à l'utilisateur de faire une recherche sur une gamme de prix correspondant au budget qu'il souhaite consacrer à un article. Dans ce projet, on implémentera et analysera ce type de recherche en se basant sur des arbres binaires de recherche, qui sont particulièrement adaptés pour ce problème.

1 Implémentation

On vous demande d'implémenter une structure d'arbre binaire de recherche dont les clés sont des `double` (*i.e.* le prix d'un objet) et dont les valeurs associées sont du texte (des tableaux de `char`s).

Fichiers fournis Les fichiers suivants vous sont fournis :

database.csv qui contient une base de données d'objets et des prix associés.

main.c qui contient un petit programme permettant de charger un fichier (*i.e.* la base de données) dans le dictionnaire.

BinarySearchTree.h qui contient l'interface à implémenter.

Interface Le dictionnaire doit répondre à l'interface suivante :

`createEmptyBinarySearchTree` qui crée un nouvel arbre binaire de recherche vide.

`createBinarySearchTree` qui, sur base de tableaux de clés et de valeurs, construit un arbre binaire de recherche équilibré (c'est-à-dire dont la hauteur est $\Theta(n \log n)$).

`freeBinarySearchTree` qui libère la mémoire allouée par la structure de l'arbre.

`getNumElements` qui retourne le nombre d'éléments dans l'arbre.

`getHeight` qui retourne la hauteur de l'arbre.

`insertElement` qui permet d'ajouter une nouvelle paire clé-valeur dans l'arbre. Si la clé existe déjà, sa valeur est remplacée par la nouvelle.

`findElement` qui retourne la valeur associée à une clé de l'arbre.

`removeElement` qui supprime une paire clé-valeur de l'arbre.

`getElementsInRange` qui retourne un tableau de valeurs dont les clés sont comprises dans un intervalle. Les valeurs doivent être triées dans l'ordre croissant de leurs clés.

2 Analyse

Dans votre rapport, on vous demande de répondre aux questions suivantes.

2.1 Analyse théorique

1. Expliquez brièvement vos choix d'implémentation.
2. En ce qui concerne la fonction `createBinarySearchTree` :
   (a) Décrivez cette fonction dans le formalisme de votre choix.
   (b) Analysez sa complexité au pire et au meilleur cas en fonction de la taille N du tableau.
3. En ce qui concerne la fonction `getElementsInRange` :
   (a) Décrivez cette fonction dans le formalisme de votre choix.
   (b) Analysez sa complexité au pire et au meilleur cas en fonction du nombres de clés R tombant dans l'intervalle et du nombre total de clés N. [1]
4. Expliquez l'intérêt d'utiliser un arbre binaire de recherche pour faire des recherches dans un intervalle plutôt que les structures suivantes :
   — Un tableau trié
   — Une table de hachage
5. Dans le cadre d'une application de réservations de vols d'avion, on souhaite pouvoir effectuer des recherches sur des intervalles de prix et de dates simultanément.
   (a) Proposez une solution raisonnablement efficace pour ce cas d'utilisation.
   (b) Analysez sa complexité au pire et au meilleur cas en fonction du nombre total de vols N, du nombre vols P dans l'intervalle de prix demandé et du nombre de vols D compris entre les deux dates fournies.

2.2 Analyse empirique

Pour les trois situations suivantes :

— Les objets sont insérés par ordre croissant de clés.
— Les objets sont insérés aléatoirement.
— Les objets sont insérés d'un bloc via `createBinarySearchTree` (*i.e.* l'arbre est parfaitement équilibré).

On vous demande de :

1. Reportez graphiquement pour des valeurs croissantes de N :
   (a) Les temps moyens d'insertion de N clés dans l'arbre.
   (b) Les temps moyens d'une recherche dans un arbre de N clés.
2. Commentez ces résultats.

1. Pour cette analyse, vous devez supposer que R et N sont donnés et supposés très grands. Vous ne pouvez donc pas par exemple définir vos meilleur et pire cas en choisissant des valeurs particulières de R et/ou N.

3 Deadline et soumission

Le projet est à réaliser **individuellement** pour le **dimanche** ~~22 novembre~~ **29 novembre 2015** à **23h59** au plus tard. Le projet est à remettre via la plateforme *cicada* :

http://submit.run.montefiore.ulg.ac.be/.

Il doit être rendu sous la forme d'une archive `tar.gz` contenant :

1. Votre rapport (5 pages maximum) au format PDF. Soyez bref mais précis et respectez bien l'ordre des (sous-)questions.
2. Le fichier `BinarySearchTree.c` répondant à l'interface de `BinarySearchTree.h`.

Vos fichiers seront évalués sur les machines ms8xx avec la commande :

```
gcc BinarySearchTree.c main.c --std=c99 --pedantic -Wall -Wextra -Wmissing-prototypes -o bst
```

Ceci implique que :

— Le projet doit être réalisé dans le standard C99.

— La présence de *warnings* impactera négativement la cote finale.

— Un projet qui ne compile pas avec cette commande sur ces machines recevra une cote nulle (pour la partie code du projet).

Un projet non rendu à temps recevra une cote globale nulle. En cas de plagiat avéré, l'étudiant se verra affecter une cote nulle à l'ensemble des projets.

Les critères de correction sont précisés sur la page web des projets.

Bon travail !

4 Annexe

4.1 Formattage de database.csv et parsing

Le fichier **database.csv** est formaté d'une manière très simple. Chaque produit est représenté par une ligne, contenant son nom, suivi du caractère `;` et du prix. Une ligne se termine par le caractère de retour à la ligne `\n`. Pour ajouter un produit, il suffit donc de rajouter une nouvelle ligne respectant ce format.

Attention si vous programmez sous Windows La plupart des éditeurs de texte (comme Bloc-notes), ajoute le caractère `\r` en plus de `\n` à chaque retour à la ligne, ce qui peut nuire au bon fonctionnement du parsing proposé dans **main.c**. Il existe toutefois des éditeurs de texte pour Windows (notamment NotePad++) qui permettent d'afficher ces caractères et de n'insérer qu'un simple `\n` à chaque retour à la ligne.

4.2 Les chaînes de caractère en C

Ce projet fait appel aux chaînes de caractères. En langage C, il n'existe pas de type `String` comme en Java ou en C++ ; à la place, on manipule des tableaux de caractères (`char[]` ou `char*`). Or, en C, il n'est pas possible de connaître la taille d'un tableau sans la conserver quelque part tout au long d'un programme, ce qui complique la manipulation de chaînes de caractères quand leur taille est variable (ce qui est le cas des noms de produits listés dans **database.csv**).

Cependant, il existe une convention pour pallier à cette limite dans le cas des chaînes de caractères : le caractère `\0` signale la fin de la chaîne de caractère. En comptant le nombre de caractères jusqu'à ce caractère final, il est donc possible d'obtenir la longueur de la chaîne.

Cette convention est utilisée systématiquement au sein de la bibliothèque `string.h` qui définit l'ensemble des fonctions de manipulation des chaînes (fusionner deux chaînes, rechercher dans une chaîne, calculer sa longueur, etc.).